Das Test Object Management Framework (TOMF)

Mario Friske
Mario Friske Consulting, Berlin

Zusammenfassung. Das Test Object Management Framework (TOMF) ermöglicht die automatisierte Orchestrierung, Konfiguration und Steuerung von Simulatoren und Hardware, um komplexe Testumgebungen vollautomatisiert bereitzustellen. TOMF wurde in einem IT-Großprojekt entwickelt und erfolgreich teststufenübergreifend eingesetzt.

Hintergrund. Die Telematikinfrastruktur (TI) soll alle Beteiligten im Gesundheitswesen wie Ärzte, Zahnärzte, Krankenhäuser, Apotheken und Krankenkassen miteinander vernetzen. Der Zugang zur TI erfolgt derzeit über einen sogenannten Konnektor. Dabei handelt es sich um ein Hardwaregerät, dass einem DSL-Router ähnelt, jedoch auf einem deutlich höheren Sicherheitsniveau arbeitet. Konnektoren werden von verschiedenen Herstellern angeboten, oftmals im Paket inklusive Kartenterminals und VPN-Zugangsdienst zur TI.

Der Betrieb sämtlicher TI-Produkte erfordert eine Zulassung durch die offizielle Zulassungsstelle. Dazu ist nachzuweisen, dass Produkte die im Produktsteckbrief gestellten Anforderungen an Interoperabilität, Sicherheit und Funktionsfähigkeit erfüllen. Je nach Produkt sind die Zulassungsaktivitäten teilweise mit erheblichen Testaufwänden der Hersteller verbunden, sodass eine weitgehende Testautomatisierung angestrebt wird.

Im Folgenden wird ein Framework zur Orchestrierung von Simulatoren für den vollautomatisierten Test auf verschiedenen Teststufen vorgestellt, dass im Kontext der erfolgreich durchgeführten Zulassungstests des Konnektors der Telekom entwickelt wurde.

Herausforderungen & Lösungsansatz. Es zeigte sich sehr schnell, dass die erforderlichen Tests komplexe und sehr unterschiedliche Testumgebungsaufbauten benötigten. Durch jeweils dedizierten Aufbau einer reinen Hardwareumgebung waren die gestellten Anforderungen nur schwer zu erfüllen.

Zu den Herausforderungen in diesem Zusammenhang zählten insbesondere die folgenden:

- Es musste ein regelmäßiger Performancetest mit bis zu 40 Kartenterminals durchgeführt werden.
- Die komplexen Use-Cases sind im Integrationstest nur End-to-End testbar, d.h. inklusive Praxisverwaltungssystem, Kartenterminals und TI.
- Es mussten verschiedene Netzwerkmodi getestet werden, d.h. serielle und parallele Anbindung.
- Für die Zertifizierung war es erforderlich, diverse Fehlersituationen herzustellen.
- Hohe Sicherheitsanforderungen der Hardwareprodukte erschweren Setup und Fehleranalyse.
- Es gab PKI-bedingte Limitierungen bei der Verwendung von Hardwarekarten.
- Es war keine automatisierte Lösung zum Bestücken und Bedienen von Hardware-Kartenterminals vorhanden.
- Neuartige und seltene Kartentypen wurden für den Test benötigt, waren aber nur schwer verfügbar.

Als Lösungsansatz bot sich der konsequente Einsatz von Simulatoren in sämtlichen Teststufen an. Deshalb wurde in dem Projekt ein dediziertes Testmittelteam aufgestellt, welches gemeinsam mit beteiligten Projektpartnern Simulatoren für sämtliche benötigten Komponenten (Karten, Kartenterminals, TI-Komponenten, Fachdienste, etc.) entwickelt hat. Werkzeuge zur Erzeugung und Nutzung einer eigenen Test-PKI wurden ebenfalls entwickelt. Weiterhin wurde ein Framework zur automatisierten Konfiguration der Testumgebung entwickelt, das Test Object Management Framework (TOMF).

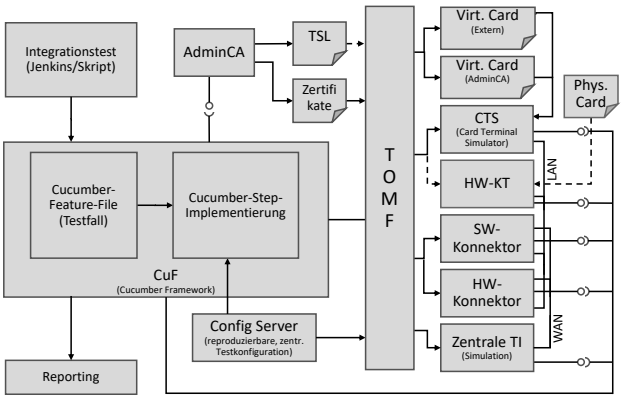

Abbildung 1: Gesamtüberblick über die Testumgebung und die mithilfe von TOMF orchestrierten Simulatoren

TOMF ermöglicht die automatisierte Orchestrierung, Konfiguration und Steuerung von Simulatoren und Hardware, wobei gemischte Konfigurationen aus Hardware und Simulationen unterstützt werden. Abbildung 1 zeigt auf der rechten Seite einige unterstützte Simulatoren, welche mittels TOMF aus dem auf der linken Seite dargestellten Testframework automatisiert orchestriert und konfiguriert werden. Alternativ ist auch ein interaktives Setup möglich.

TOMF basiert auf frei verfügbarer Software, wie Java, Git, Maven, Nexus, Docker, Bouncy Castle, etc.

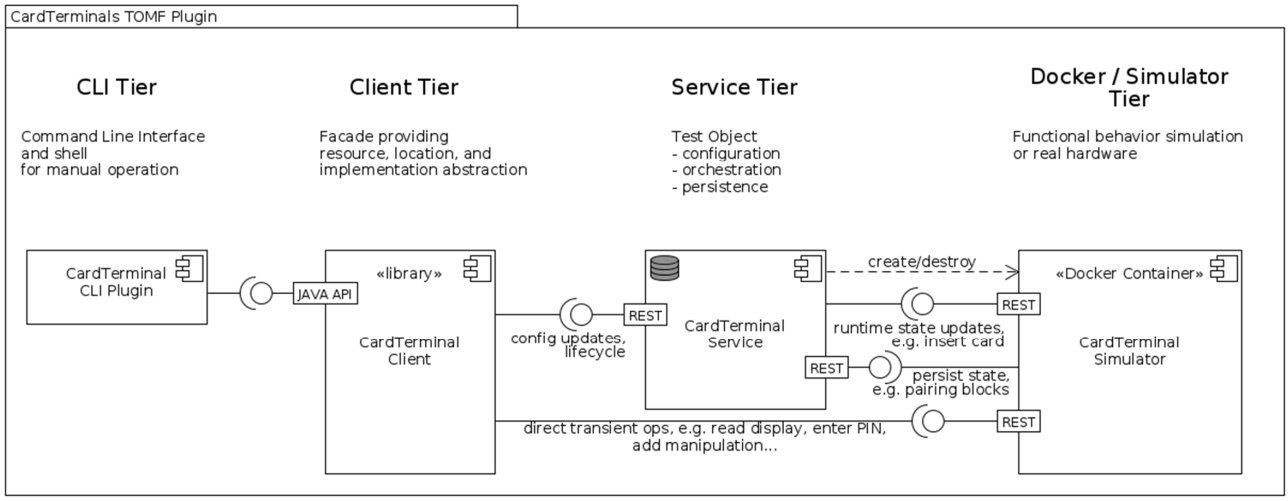

Abbildung 2: Technischer Aufbau eines Simulators am Beispiel des Kartenterminalsimulators

Das Framework nutzt Docker als Infrastruktur zur Containervirtualisierung. Projektintern wurde TOMF via Source Code und Dockerimages verteilt. Wesentliches Merkmal von TOMF ist eine erweiterbare Pluginorientierte Microservice-Architektur, in welcher für jeden einzubindenden Simulator ein domänenspezifisches PlugIn zu erstellen ist.

Abbildung 2 zeigt am Beispiel des Kartenterminalsimulators den prinzipiellen Aufbau eines Simulator-Plugins. Im hinten liegenden Simulator- bzw. Docker-Tier wird der eigentliche Simulator als Docker-Container bereitgestellt. Der davor liegende Service-Tier steuert den Lebenszyklus des Simulators und das Konfigurieren des Simulators, im Beispiel des Kartenterminalsimulators kann das beispielsweise das Stecken und Ziehen simulierter Karten sein. Die nächste Ebene bildet der Client-Tier, welcher ein Java-API für die Automatisierungs-Frameworks anbietet. Der ganz vorne liegende CLI-Tier nutzt ebenfalls dieses Java-API, um ein CLI für die interaktive Nutzung bereitzustellen. Das CLI hat sich sowohl bei der interaktiven Entwicklung und Erprobung von Testfällen als auch bei der Fehleranalyse als sehr hilfreich gezeigt.

Abbildung 3 zeigt die logische Architektur von TOMF. Unten liegt das Basis-Framework, welches prinzipiell unabhängig von der Anwendungsdomäne ist. Darüber liegen die domänenspezifischen Plugins zur Ansteuerung der einzelnen Simulatoren und sonstigen Komponenten. Ganz oben liegt die TOMF-Runtime welche zwar prinzipiell domänenunabhängig ist, jedoch eine spezifische aufeinander abgestimmte Kombination von Plugins in Form eines Releases für eine domänenspezifische Anwendung bereitstellt. Im Konnektor-Projekt wurde auf diese Art und Weise ein spezifisches TOMF-Release für jede Konnektor-Produktlinie bereitgestellt.

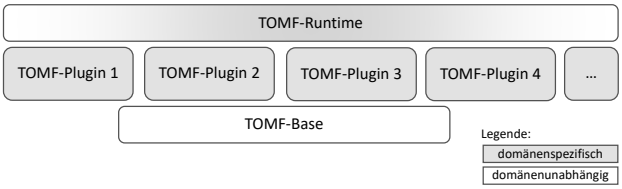

Abbildung 3: Logische Architektur von TOMF

Status & Ausblick. Mit TOMF wurde ein allgemeiner Lösungsansatz entwickelt, der immer dann eingesetzt werden kann, wenn komplexe Integrationstestumgebungen (Simulatoren & Hardware) vollautomatisiert, reproduzierbar und skalierbar provisioniert werden müssen.

Im Kontext des Konnektorprojekts wurden für sämtliche entwickelte Simulatoren TOMF-Plugins bereitgestellt. Mit deren Hilfe wurde TOMF in mehreren Teststufen (Performancetest, Integrationstest, Subsystemtest, Entwicklertests) erfolgreich zum automatisierten Provisionieren und Konfigurieren komplexer Testumgebungen eingesetzt. TOMF wurde bereits mit verschiedenen Testautomatisierungsframeworks, wie Cucumber, JUnit und Performance-Skripten, eingesetzt.

Die Eigenentwicklung des Hardware-Konnektors der Telekom ist mittlerweile abgeschlossen. Es gibt Überlegungen, den domänenunabhängigen Teil von TOMF als Open-Source-Software zu veröffentlichen.